Using the Spiral Model and MBASE to Generate New Acquisition Process Models: SAIV, CAIV, and SCQAIV

Dr. Barry Boehm, Dr. Dan Port, LiGuo Huang, and Winsor Brown
University of Southern California

In this article, we show how you can use the MBASE process framework to generate a family of acquisition process models for delivering user-satisfactory systems under schedule, cost, and quality constraints. We present the six major steps of the Schedule/Cost/Schedule-Cost-Quality as Independent Variable (SAIV/CAIV/SCQAIV) process using SAIV and a representative Department of Defense (DoD) Command, Control, and Communications Interoperability application as context. We then summarize our experience in using SAIV on 26 University of Southern California electronic services projects, followed by discussions of SAIV/CAIV/SCQAIV application in the commercial and defense sectors, of model application within the DoD acquisition framework, and of the resulting conclusions.

A number of Department of Defense (DoD) organizations are responding to the DoD Evolutionary Acquisition Initiative in DoDI 5000.2 [1] by organizing evolutionary increments of capability around the objective of developing and fielding each increment within a fixed schedule (frequently 18 or 24 months) or fixed budget. Examples are new capabilities or major upgrades for such software-intensive systems as Command, Control, and Communications Interoperability (C3I), logistics, or combat platform electronics suites.

The usual approach for achieving this objective follows this pattern:

1. Determine the best-possible set of features that can be developed and fielded within the available schedule and/or budget.
2. Contract to develop and field this feature set within the available schedule and/or budget.
3. Monitor the contractor's progress in achieving the objectives within the schedule and/or budget.

This is the usual interpretation of DoD's current Cost as Independent Variable (CAIV) approach. Unfortunately, step four of this scenario usually involves finding that the available schedule and/or budget are insufficient, and that the existing contract constraints and architectural commitments preclude finding a way to field an acceptable capability within the available schedule and/or budget.

Is this really the usual outcome? Sadly, yes, both in government and commercial software acquisition. For example, the Standish Report [2] found that 84 percent of the software-intensive system projects it surveyed either overran their budgets and schedules or were cancelled before completion. The average overruns on these projects were 189 percent of planned cost and 222 percent of planned schedule. The completed overrun projects delivered an average of only 61 percent of the originally specified features. The Standish Report does not address the effect on delivered software quality, but our analysis of similar projects indicates similar problems with delivered defect density (nontrivial defects per function point or per thousands of source lines of code).

> *"... 84 percent of the software-intensive system projects it [Standish] surveyed either overran their budgets and schedules or were cancelled ..."*

The Standish Report's and our analyses of the major root causes of this problem are:

- Schedule and budget estimates can be (sometimes wildly) optimistic.
- Even if "most likely" estimates are used, the definition of most likely means that they will be overrun in roughly half of the projects.
- To maximize the probability of successful delivery, the contractor will often use a point-solution architecture to accommodate the specified features. When the inevitable threat, combat platform, feature priority, or technology changes come, they are hard to accommodate within the point-solution architecture.

Using SAIV, CAIV, and SCQAIV as Process Models

In our earlier CrossTalk articles on the Spiral Model [3] and Model-Based (System) Architecting and Software Engineering (MBASE) [4], we showed that these were actually process model generators for the acquisition of software intensive systems. They use risk considerations to determine the most appropriate sequence of activities to perform (among specification, prototyping, simulation, benchmarking, increments of development, etc.) in order to achieve the most cost-effective system capability within various resource constraints such as cost, schedule, personnel, and platform characteristics.

In this article, we show how you can use the MBASE process framework to generate a particularly attractive family of acquisition process models for delivering user-satisfactory systems under schedule, cost, and quality constraints.

The risk-driven MBASE-Spiral approach uses the risk of schedule or cost overrun to invert the usual software-intensive-system acquisition process. Either schedule, cost, or some combination of schedule, cost, and quality becomes the independent variable, and the lower-priority features become the dependent variable. This requires several sub-processes:

- Determination of a top-priority core capability and quality level strongly assured to be achievable within the schedule-cost-quality constraints.
- User expectations management and continuing update of feature priorities.
- Architecting the system for ease of dropping borderline-priority features and future addition of lower-priority features.
- Careful progress monitoring and corrective action to keep within cost-schedule-quality constraints.

In this article, we next present the six major steps of the Schedule/Cost/Schedule-Cost-Quality as Independent Variable (SAIV/CAIV/SCQAIV) process

using SAIV and a representative DoD C3I application as context. We then summarize our experience in using SAIV on 26 University of Southern California (USC) electronic services projects, 24 of which have successfully delivered systems with high client-satisfaction ratings on a fixed schedule. This is followed by discussions of SAIV/CAIV/SCQAIV application in the commercial and defense sectors, of model limitations and extensions, and of the resulting conclusions.

The SAIV Process Model

The key to successful SAIV practice is to strategically plan through all life-cycle areas to meet a delivery date. SAIV is defined by explicitly enacting the following six process elements:

1. Manage expectations by establishing a stakeholders' shared vision of achievable objectives.
2. Prioritize system features.
3. Estimate subsets of features that can be developed with high confidence within the available schedule.
4. Establish a coherent set of core capabilities with borderline features to be added if possible, and a software/system architecture to easily accommodate borderline features.
5. Plan development increments, including a high-confidence core capability and next-priority subsets.
6. Execute development plans with careful change and progress monitoring and control processes.

The MBASE process model generator is used to generate a SAIV process model suitable for a particular project. Figure 1 shows the SAIV version of the Win-Win Spiral Model. The process models for CAIV and SCQAIV are essentially the same except for the definition of the radial dimension of the spirals. For CAIV projects, the spiral's traditional radial dimension of cumulative cost is used. For SAIV projects, the radial dimension is cumulative calendar time. Either cost or time can be used for SCQAIV, with the other objective and desired quality acting as constraints.

Figure 1 also shows the major SAIV process elements to be described next. These are executed concurrently within the spirals. As discussed in our updated spiral model article [3], feedback and iteration of previous-cycle results are part of the spiral process, but are omitted from Figure 1 for simplicity.

The milestone content and pass-fail criteria for the Life Cycle Objectives (LCO), Life Cycle Architecture (LCA), and Initial Operational Capability (IOC) in

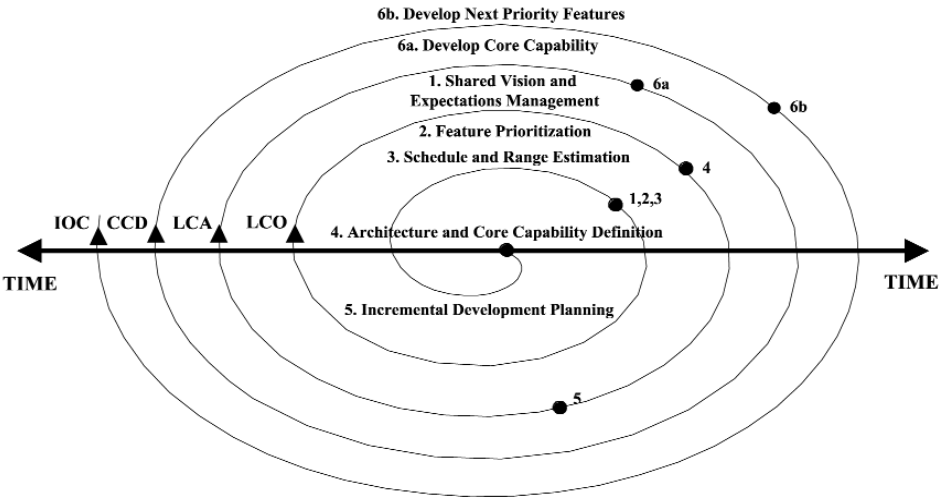

Figure 1: *Mapping of SAIV Spiral Process Elements Onto Win-Win Model*

Figure 1 were described in detail in December CrossTalk's article on the Spiral Model and MBASE [4]. They are also the major development milestones in the Rational Unified Process [5, 6]. We will elaborate the LCA milestone content in a following section. The SAIV/CAIV/SCQAIV family of process models adds a further milestone in Figure 1: the Core Capability Demonstration (CCD). It will be detailed, too, in later sections.

A Representative C3I System

We now elaborate and illustrate the six SAIV steps in the context of a representative C3I system. The current system has three major upgrade requirements: changing to a Web-based operation; changing to an XML-based interoperability scheme; and adding a new weather-impact capability to support better operational planning, task planning, and battle management decision making. A new fielded capability is needed in 19 months to maintain compatibility with other interoperating systems transitioning to the Web and XML at that time.

Shared Vision and Expectations Management

As graphically described in Death March [7], many software projects lose the opportunity to assure a rapid, on-time delivery by inflating client expectations and over promising on delivered capabilities. The first step in the SAIV process model is to avoid this by obtaining stakeholder agreement that meeting a fixed schedule for delivering the system's IOC is the most critical objective, and that the other objectives such as the IOC feature content can be variable, subject to meeting acceptable levels of quality and post-IOC scalability.

For the example C3I system, the 19-month IOC milestone is clearly critical for interoperability. Early meetings of the system's integrated product team should emphasize that meeting this milestone may be incompatible with stakeholders getting all the features they want.

Feature Prioritization

With MBASE at USC, stakeholders use the USC/GroupSystems.com EasyWinWin requirements negotiation tool [8] to converge on a mutually satisfactory (win-win) set of project requirements. One step in this process involves the stakeholders prioritizing the requirements by assessing their relative importance and difficulty, each on a scale of zero to 10. This process is carried out in parallel with initial system prototyping, which helps ensure that the priority assessments are realistic.

Easy WinWin has been used successfully for DoD software applications [9]. However, other collaboration tools or even manual group-meeting techniques can be used for this step. In our C3I example, the stakeholders rate the Web and XML capabilities higher-priority based on interoperability essentials, but agree that Weather capabilities are important also.

Schedule Range Estimation

The developers then use a mix of expert judgement and parametric cost modeling to determine how many of the top-priority features can be developed in 24 weeks under optimistic and pessimistic assumptions. For the parametric model, we use Constructive Cost Model (COCOMO) II, which estimates 90 percent confidence limits on both cost and schedule [10]. Other models such as Software Life-Cycle Model (SLIM) [11], System Evaluation and Estimation of Resources (SEER) [12], and Knowledge PLAN [13] provide simi-

Capabilities	Fastest Achievable Schedule (months)	
	Most Likely (50% confidence)	90% confidence
Either Web or XML	12	15
Both Web and XML	15	19
Web, XML, and Weather	19	24

Table 1: *Fastest Achievable Schedules for C3I Capabilities*

lar capabilities.

Table 1 summarizes the results of a COCOMO II analysis of the example C3I system. It shows the fastest achievable schedules for completing either the Web or XML capabilities (each require 12 months at best); both the Web and XML; or all three capabilities (Weather requires 14 months at best). The two columns show the most likely schedule (achievable 50 percent of the time) and the 90-percent confidence schedule (achievable 90 percent of the time).

The stakeholders see that all three capabilities can be achieved in 19 months in the most likely estimate, but are concerned that this means that the 19-month schedule will be overrun about half the time; furthermore, that with 90 percent confidence it will take up to 24 months, an unacceptable outcome. However, the Web and XML capabilities could be completed in 19 months 90 percent of the time.

Architecture and Core Capability Determination

The most serious mistake a project can make at this point is just to pick the top-most priority features with 90 percent confidence of being developed in 19 months. This can cause two main problems: producing an IOC with an incoherent and incompatible set of features, and delivering these without an underlying architecture supporting easy scalability up to the full feature set and workload.

First, the core capability must be selected so that its features add up to a coherent and workable end-to-end operational capability. Second, the remainder of the lower-priority IOC requirements and subsequent evolution requirements must be used in determining a system architecture facilitating evolution to full operational capability. Still the best approach for achieving this is to use the Parnas information-hiding approach to encapsulate the foreseeable sources of change within modules [14]. The architecting process may take two or more win-win spiral cycles of prototyping, commercial off-the-shelf (COTS) product evaluation, and stakeholder renegotiation to reconcile the system's product, process, property, and success models into a LCA package.

The C3I system stakeholders determine that the core capability should include the critical subsets of the Web, XML, and Weather capabilities, rather than all of the Web and XML capabilities. This is both because the Weather decision support is much needed, and because it would be infeasible to add a significant Weather capability in just the time left after the core capability was completed.

Incremental Development Planning

The LCA package includes an incremental development plan (item 5 in Figure 1, page 21) indicating the schedules and pass/fail criteria for the core capability (item 6a), IOC (item 6b), and perhaps other milestones.

Since the core capability has only a 90 percent assurance of being completed in 19 months, this means that about 10 percent of the time, the project will have to stretch to deliver the core capabilities in 19 months, perhaps with some performer overtime or completion bonuses, or occasionally by further reducing the top-priority feature set. In the most likely case, however, the project will achieve its core capability with about 20 percent to 30 percent of the schedule remaining. This time can then be used to add the next-highest priority features into the IOC (again, assuming that the system has been architected to facilitate this).

An important step at this point is to provide the operational stakeholders (users, operators, maintainers) with a core capability demonstration. Often, this is the first point at which the realities of actually taking delivery of and living with the new system hit home, and their priorities for the remaining capabilities may change.

Also, this is an excellent point for the stakeholders to reconfirm the likely final IOC content, and to synchronize plans for conversion, training, installation and cutover from current operations to the new IOC.

Development Execution; Change and Progress Monitoring and Control

As progress is being monitored with respect to plans, there are three major sources of change that may require reevaluation and modification of the project's plans:

1. Schedule slips. Traditionally, these can happen because of unforeseen technical difficulties, staffing difficulties, customer or supplier delays, etc.
2. Requirements changes. These may include changes in priorities, changes in current requirements, or needs for new high-priority requirements.
3. Project changes. These may include staffing changes, COTS changes, or new marketing-related tasks (e.g., interim sponsor demos).

In some cases, these changes can be accommodated within the existing plans. If not, there is a need to rapidly renegotiate and restructure the plans. If this involves the addition of new tasks on the project's critical path, some other tasks on the critical path must be reduced or eliminated. There are several options for doing this, including dropping or deferring lower-priority features, reusing existing software, or adding expert personnel. In no cases should new critical-path tasks be added without adjustments in the delivery schedule or other schedule drivers.

By following these guidelines, the C3I project should be able to overcome the usual sources of change above and successfully deliver a core capability within 19 months, often with most of the full set of capabilities added as well. However, although SAIV can significantly improve your success rate, it can't guarantee success in all situations, such as major budget cuts or radical project redirections. In these cases, budget, schedule, and core capability content will need to be significantly renegotiated. Some examples of failed SAIV projects are given in the next section.

SAIV Experience

The SAIV process model is described in terms of a representative set of SAIV applications: USC's annual series of real-client campus e-services projects [15, 16]. These projects are largely Web-based applications developed by five-person master's-student teams, using the MBASE guidelines [17] and the MBASE Electronic Process Guide [18].

The teams' main challenges are to develop a LCO package and a LCA package, described below, for a USC

Information Services Division client's application in 12 weeks during the fall semester, and to develop and transition an IOC in 12 weeks during the spring semester. These are extreme examples of schedule being the independent variable since the USC semester schedule is fixed and the students disappear (to graduation or summer jobs) at the end of the spring semester.

The critical success factors of the MBASE process model involve the concurrent development of several initial artifacts: an operational concept description, a requirements definition, an architecture description, a life-cycle plan, a feasibility rationale, and one or more prototypes. These are evaluated at two major pass/fail points, the LCO and the LCA milestones. Both milestones use the same primary pass-fail criterion: If we build the system to the given architecture, it will satisfy the requirements, support the operational concept, be faithful to the prototypes, and be buildable within the processes, budgets, and schedules in the plan.

For the LCO milestone, this criterion must be satisfied for at least one choice of architecture, along with demonstration of a viable business case for the system and the expressed concurrence of all the success-critical stakeholders. For the LCA milestone, the pass-fail criterion must be satisfied for the specific choice of architecture and COTS components to be used for the system, along with continued business case viability and stakeholder concurrence, plus elimination of all major project risks or coverage of the risks in a risk management plan.

One of our primary goals in the project course is to give the students experience in risk management [19]. Our risk management lectures and homework exercises emphasize a list of the 10 most serious risk items: Personnel risks are number one, and budget-schedule risks are number two. The student projects' risk management plans must show how their team will avoid the risks of delivering an unsatisfactory LCA package in the first 12 weeks (fall semester), and of unsatisfactorily delivering and transitioning an IOC in the second 12 weeks (spring semester). The MBASE guidelines recommend that they adopt the SAIV model described in an earlier section; so far, all the projects have done this.

Also, we work in advance with the USC e-services clients to sensitize them to the risks of over-specifying their set of desired IOC features, and to emphasize the importance of prioritizing their desired capabilities. This generally leads to a highly collaborative win-win negotiation of prioritized capabilities, and subsequently to a mutually satisfactory core capability to be developed as a low-risk minimal IOC.

The projects' monitoring and control activities include the following:

- Development of a top-N project risk item list that is reviewed and updated weekly to track progress in managing risks (N is usually between five and 10).
- Inclusion of the top-N risk item list in the project's weekly status report.
- Management and technical reviews at several key milestones.
- Client reviews at other client-critical milestones such as the core capability demonstration.

The use of SAIV and these monitoring and control practices have led to on-time, client-satisfactory delivery and transition of 24 of the 26 products developed to date. One of the two failures was in our first year, when we tried to satisfy three clients by merging their image archive applications into a single project; we underestimated the complexity of the merge. As a result, "merging multiple applications" has become one of the major sources of project risk that we consider.

The second failure happened recently when a project that appeared to be on track at its transition readiness review, simply did not implement its transition plan when its client suddenly had to go out of town. We were not aware of this until the client returned after the semester was over and the students had disappeared to graduation and summer jobs. We have since revised our system of closeout reviews to eliminate this "blind spot" and related problem sources.

On the other 24 projects, client evaluations have been uniformly quite positive, averaging about 4.4 on a scale of one to five. A particularly frequent client evaluation comment has been their pleasure in being able to synchronize product transition on a specific fixed date with their other transition activities. Another pleasant surprise was the effect on clients' review timeliness: "You mean if I evaluate the prototype right away, I'll get more features in my IOC?" The e-services project artifacts can be reviewed on the class Web page <http://sunset.usc.edu/classes>.

E-Commerce Projects

One of our industrial affiliates, C-Bridge, Inc., uses a very similar SAIV process model, which enables them to consistently deliver e-commerce systems on fixed schedules between 13 and 26 weeks. Their Rapid Value approach uses milestones very similar to MBASE's LCO, LCA, and IOC milestones; their counterpart phases are named define, design, develop, and deploy. They use similar approaches in working in advance with their clients to ensure a workable SAIV scope and schedule, and in anticipating and pre-working potential transition problems to client-based operations and maintenance [20].

The CAIV and SCQAIV Process Models

Simply substituting "cost" for "schedule" in the SAIV process model described above provides you with an equally effective way to use CAIV as a process model. The SCQAIV model is a straightforward extension of CAIV and SAIV. It involves setting the system's quality goals (e.g., a delivered defect density of 0.3 nontrivial defects per thousand source lines of code (KSLOC), or of 0.03 nontrivial defects per function point), and tracking progress with respect to achieving the desired combination of schedule, cost, or quality goals.

If any of these goals becomes unachievable in delivering the current feature set, the project must drop enough lower-priority features to make the combination of goals achievable. There may be limits to the project's ability to do this, such as insufficient schedule to develop even a viable core capability. We have discussed this situation via a production-function perspective in [21].

DoD Acquisition Framework

In situations such as post-deployment upgrades and pre-planned product improvements, DoD can and often has implemented versions of SAIV/CAIV/SCQAIV as smoothly as they are done commercially. Frequently, in such situations, the organization's software maintenance budget and release cycle are relatively fixed. The biggest risk is to promise too much within these constraints, leaving the sacrifice of quality as the only way to meet budget and schedule. This inevitably leads to degradations of the software's maintainability, operational fitness, and future maintenance productivity.

Thus, a form of SCQAIV is the best option for software maintenance in which quality standards are set, infrastructure upgrades are given appropriate priorities, and lower-priority features are shed to meet cost, schedule, and quality objectives.

This approach is workable because

DoD's operations and maintenance acquisition practices are similar to their commercial counterparts. Budgets are generally not tied to premature promises of delivered features, and there is usually a long-term customer-supplier relationship with a shared product vision among the customer, supplier, and users. This continuing relationship usually increases mutual trust, the ability to share and respect each other's win conditions, and the ability to negotiate and, when necessary, readily adjust mutually satisfactory or win-win agreements and priorities.

Competitive Development

In some cases, such as the Air Force Electronic Systems Center's Command Center Product Line (CCPL) and within classified-application organizations, DoD customers have been able to create development arrangements similar to the stable post-deployment support situation described above. CCPL, for example, developed a flexible contractual instrument focused on creating user value rather than pre-specified features, and allowing in-process renegotiation of priorities. It selected three contractors via competitive source selection. The evaluation criteria included track record on similar developments, software Capability Maturity Model® (CMM®) process maturity, technical and management approach, and demonstration of the approach via a representative exercise.

Once selected, the contractors operated as a team with the customer, developing a strong shared vision for the product line, and taking on new assignments based on best-matched available expertise, ensuring effective employment of all three contractors' resources. In this situation, SAIV/CAIV/SCQAIV-type approaches were highly feasible and preferable.

In many cases, however, DoD organizations must develop a new system vision and set of acquisition parameters (schedule, cost, quality attribute levels, and feature scope) within a competitive acquisition framework. Here, complete multi-contractor shared vision development is impractical, as developers will be unwilling to share their competitive-discriminator technology solutions with competing developers.

Frequently, this leads acquisition organizations to exclude developers from participating in the creation of the shared vision. This is highly risky, as the resulting decision may exclude attractive developer technology solutions. It may also leave serious vision mismatches between the customer, user, and selected developers, making SAIV/CAIV/SCQAIV system scoping and feature prioritization difficult to achieve, particularly if the program's funding and schedule have been tied to a particular set of delivered capabilities.

Unfortunately, there is no ideal solution to this dilemma. The most attractive near-solution involves the use of multiple competitive spiral cycles of system definition, with the number of competitors being reduced from one cycle to the next. The earlier cycles are shorter and less expensive, making a larger number of participants affordable. They can be run as SAIV/CAIV procurements with equal opportunity for each competitor. Some care is necessary to avoid leaking competitors' key discriminators, but many competitive DoD concept definition efforts have achieved this.

"The six-step SAIV process model presented here has been used successfully on 24 of 26 e-service applications at USC ..."

These earlier cycles enable overall system scoping and tradeoff analysis to be performed, along with the evaluation of readiness levels of key technologies via prototyping, benchmarking, modeling and simulation, etc. This also enables the acquirers to evaluate the competing developers' capabilities and understanding of the system context and objectives.

Some similar criteria to those used by CCPL (track record, technical and management capabilities, concept definition and evaluation performance) are used for initial competitor selection and early down-selection. Later down-selection criteria increasingly involve development capabilities such as process maturity and realism of development plans, schedules, and budgets. Here again, the final development competition can fix the cost and/or schedule, provide a prioritized feature set, and compete on scope and realism of feature set delivery plans.

All three services are making progress toward mastering this kind of evolutionary acquisition in the context of the new DoD 5000-series of acquisition regulations [1]. These include the extensive use of simulation and modeling in the Army's SMART program, the Air Force's Instruction 63-123, evolutionary acquisition of command and control systems [22], use of downselected contractors' expertise in other system life-cycle roles, and service use of new contractual vehicles such as cooperative research and development agreements (CRADAs) and the Defense Advanced Research Projects Agency (DARPA)-originated "other transactions" approach [22]. All of these are compatible with and have been used successfully with SAIV/CAIV/SCQAIV-type approaches.

Conclusions

The six-step SAIV process model presented here has been used successfully on 24 of 26 e-services applications at USC, and on a similar percentage of e-commerce applications at C-Bridge, to deliver highly client-satisfactory applications on a fixed schedule in a climate of rapid change. Its 92 percent success rate compares favorably with the 16 percent success rate in the Standish Group's survey of current practice. There might be some concern that student-team projects are easier than professional-developer projects, but there is also a case that on-schedule delivery is harder with teams of people who are unfamiliar with each other, with project practice, with the clients, and with the applications domain (all generally true for the USC e-services projects).

The critical success factors of the SAIV approach are:

- Working with stakeholders in advance to achieve a shared product vision and realistic expectations.
- Getting clients to develop and maintain prioritized requirements.
- Scoping the core capability to fit within the high-payoff segment of the application's production function for the given schedule.
- Architecting the system for ease of adding and dropping borderline features.
- Disciplined progress monitoring and corrective action to counter schedule threats.

The approach can also be applied to its counterpart CAIV process model. It can also provide a way to transform the current dilemma, "Schedule, Cost, Quality: Pick Any Two," to "Schedule, Cost, Quality: Pick All Three," via the SCQAIV version of the model whenever your project is able to shed lower-priority features to meet its SCQ objectives. Proven

strategies are also available for applying SAIV/CAIV/SCQAIV to competitive DOD system acquisitions.◆

References

1. Department of Defense Instruction 5000.2. Operation of the Defense Acquisition System. Sept. 2000, <www.acq.ord.mil/ap/i50002p.doc>.
2. Standish Group. "CHAOS," <www.standishgroup.com/chaos.htm>.
3. Boehm, B., and W. Hansen. "The Spiral Model as a Tool for Evolutionary Acquisition," CROSSTALK, May 2001: pp. 3-11.
4. Boehm, B., and D. Port. "Balancing Discipline and Flexibility with the Spiral Model and MBASE," CROSSTALK, Dec. 2001.
5. Kruchten, P. The Rational Unified Process. Addison-Wesley, 1998.
6. Royce, W.E. Software Project Management: A Unified Framework. Addison-Wesley, 1998.
7. Yourdon, E. Death March. Prentice Hall, 1997.
8. Boehm, B., P. Gruenbacher, and R. Briggs, eds. "Developing Groupware for Requirements Negotiation: Lessons Learned," IEEE Software. May/June 2001: pp. 46-55.
9. Saboe, M., P. Gruendacher, and P. Kloska. "Experience with the WinWin Process for Planning Software Infrastructure and Technology," Proceedings, International Conference of Good Systems, 2002 (to appear).
10. Boehm, B., C. Abts, A.W. Brown, S. Chulani, B.Clark, E. Horowitz, R.Madachy, D. Reifer, and B. Steece, eds. Software Cost Estimation with COCOMO II. Prentice Hall, 2000.
11. Putnam, L. "Software Life Cycle Model (SLIM)," QSM, 2001, <www.qsm.com>.
12. Galorath, D. SEER-SEM. Galorath, Inc., 2001,<www.galorath.com>.
13. Jones, C. "Knowledge PLAN," Artemis/SPR, 2001, <www.spr.com>.
14. Parnas, D., "Designing Software for Ease of Extension and Contraction," IEEE Trans. Software Engr. Mar. 1979: pp. 128-137.
15. Boehm, B., A. Egyed, J. Kwan, D. Port, A. Shah, and R. Madachy, eds. "Using the WinWin Spiral Model: A Case Study." IEEE Computer, July 1998: pp. 33-44.
16. Boehm, B., A. Egyed, D. Port, A. Shah, J. Kwan, and R. Madachy, eds. "A Stakeholder Win-Win Approach to Software Engineering Education," Annals of Software Engineering, Apr. 1999.
17. Boehm, B., D. Port, M. Abi-Antoun, and A. Egyed, eds. Guidelines for Model-Based Architecting and Software Engineering (MBASE). Ver. 2.2. USC-CSE, Feb. 2001, <http://suset.usc.edu/Research/MBASE>.
18. Mehta, N., MBASE Electronic Process Guide. USC-CSE, Sept. 1999, <http://sunset.usc.edu/Research/MBASE>.
19. Port, D. and B. Boehm, eds. "Educating Software Engineering Students to Manage Risk," Proceedings, ICSE 2001, May 2001.
20. Madachy, R., A. Pan, and A. Arboleda, eds. "Processes for Rapid Development of Internet Applications," LA SPIN Presentation, 24 Jan. 2001.
21. Boehm, B., and A.W. Brown, eds. "Mastering Rapid Delivery and Change with the SAIV Process Model," Proceedings, ESCOM2001, Apr. 2001.
22. U.S. Air Force Instruction 63-123, "Evolutionary Acquisition for C2 Systems," Apr. 2000, <http://afpubs.hq.af.mil>.

About the Authors

Barry Boehm, Ph.D., is the TRW professor of software engineering and director of the Center for Software Engineering at the University of Southern California. He was previously in technical and management positions at General Dynamics, Rand Corp., TRW, the Defense Advanced Research Projects Agency, where he managed the acquisition of more than $1 billion worth of advanced information technology systems. Dr. Boehm originated the spiral model, the Constructive Cost Model, and the stakeholder win-win approach to software management and requirements negotiation.

University of Southern California
Center for Software Engineering
Los Angeles, CA 90089-0781
Phone: (213) 740-8163
Fax: (213) 740-4927
E-mail: boehm@sunset.usc.edu

Daniel Port, Ph.D., is a research assistant professor of Computer Science and an associate of the Center for Software Engineering at the University of Southern California. He received a doctorate degree from the Massachusetts Institute of Technology, and a bachelor's degree from the University of California, Los Angeles. His previous positions were assistant professor of Computer Science at Columbia University, director of Technology at the USC Annenburg Center EC2 Technology Incubator, co-founder of Tech Tactics, Inc., and a project lead and technology trainer for NeXT Computers, Inc.

University of Southern California
Center for Software Engineering
Los Angeles, CA 90089-0781
Phone: (213) 740-7275
Fax: (213) 740-4927
E-mail: dport@sunset.usc.edu

LiGuo Huang is a Ph.D. student and research assistant in the computer science department at the Center for Software Engineering at the University of Southern California (USC). She received a bachelor's degree in engineering management from HoHai University, China, and a master's degree in computer science from USC. Her current research interests focus on software system's Schedule/ Cost/Schedule-Cost-Quality as Independent Variable process model and economic-driven software engineering. She was an Honored Student of JiangSu Province awarded by the Education Committee of JiangSu Province, China, an Honored Student of Hohai University, and was awarded the Outstanding Academic Achievement Award at USC.

University of Southern California
Computer Science Department
Los Angeles, CA 90089
Phone: (213) 740-6505
Email: liguohua@usc.edu